Assignment - 2
Data Visualization and Data Preprocessing

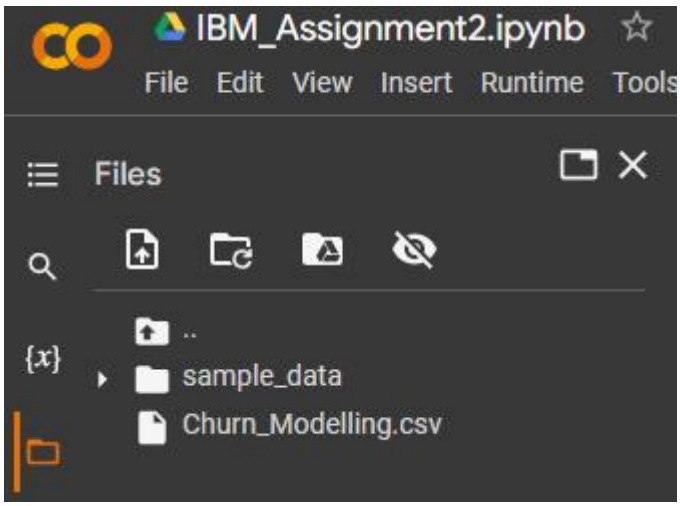
Assignment Date	21.09.2022
Student Name	Damodharan R
Student Roll Number	2019115028
Maximum Marks	2 Marks

Task-1:

Download the dataset.

Task-2:

Load the dataset.



Solution:

```
df = pd.read_csv("Churn_Modelling.csv")
df.head()
#-----------------#
#-----------------#
```

Screenshot:

```
In [116…
df = pd.read_csv("Churn_Modelling.csv")
df.head()
```

Out[116…

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Task-3:

Perform Univariate, Bivariate, Multivariate analysis.

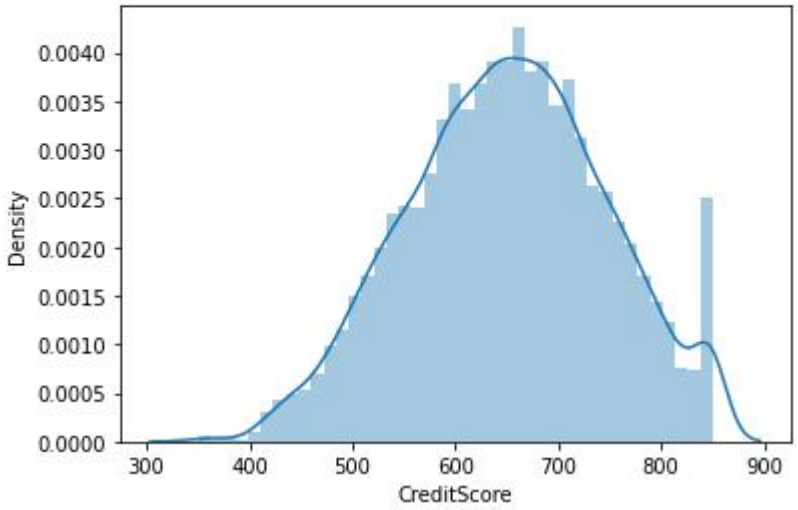
Univariate analysis solution:

```
sns.distplot(df['CreditScore'])
#-----------------#
#-----------------#
```

Screenshot:

```
In [120…  sns.distplot(df['CreditScore'])

Out[120…  <matplotlib.axes._subplots.AxesSubplot at 0x7f4f7a7be8d0>
```



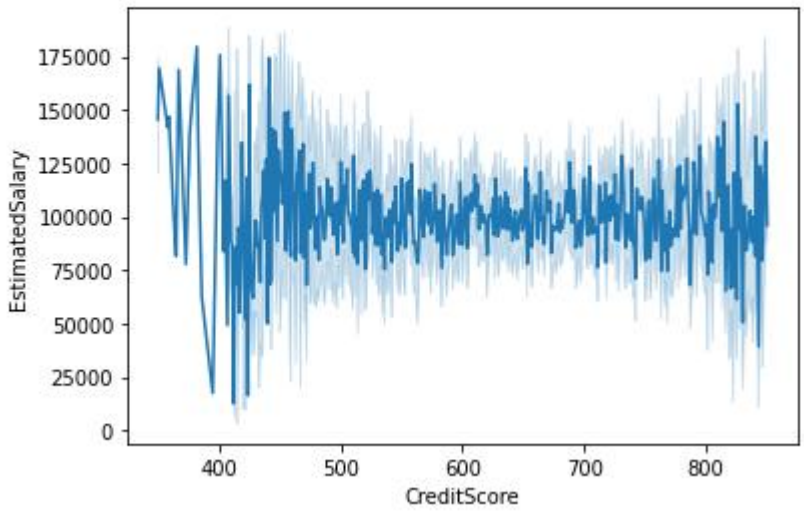
Bivariate analysis solution:

```
sns.lineplot(df['CreditScore'],df['EstimatedSalary'])
```

Screenshot:

```
In [121…  sns.lineplot(df['CreditScore'],df['EstimatedSalary'])

Out[121…  <matplotlib.axes._subplots.AxesSubplot at 0x7f4f7a6d4510>
```

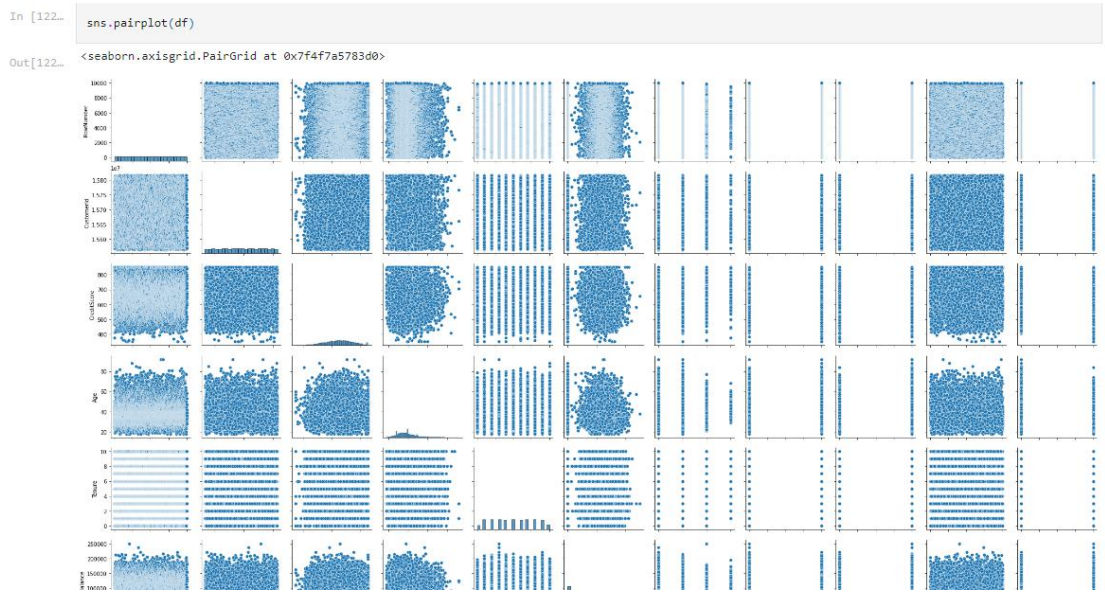


Multivariate analysis solution:

```
sns.pairplot(df)
```

#----------------#
#----------------#

Screenshot:



Heatmap solution:

sns.heatmap(df.corr())

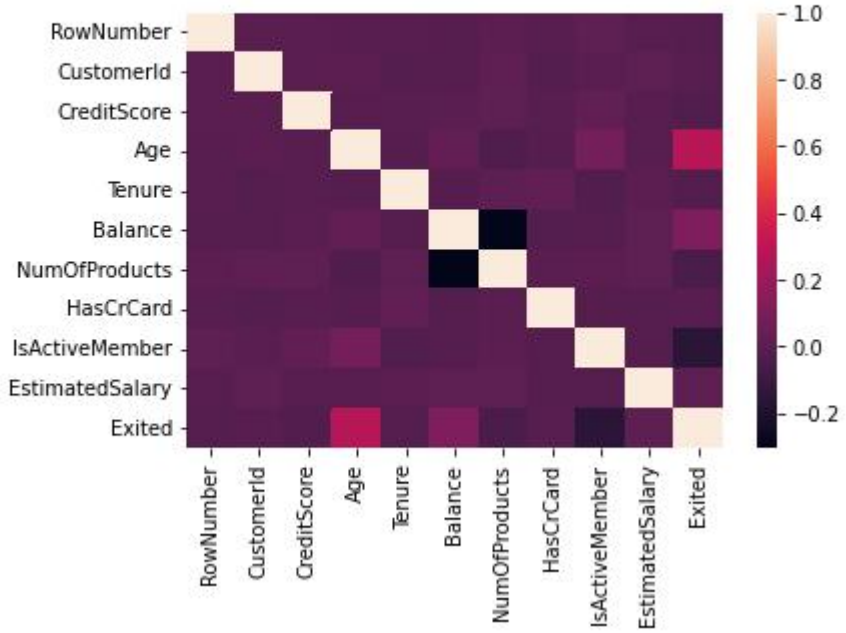
#----------------#

#----------------#

Screenshot:

```
In [125…  sns.heatmap(df.corr())

Out[125…  <matplotlib.axes._subplots.AxesSubplot at 0x7f4f773e4190>
```



Correlation solution:

df.corr()

#----------------#

#----------------#

Screenshot:

```
In [124…  df.corr()
```

Out[124…

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
RowNumber	1.000000	0.004202	0.005840	0.000783	-0.006495	-0.009067	0.007246	0.000599	0.012044	-0.005988	-0.016571
CustomerId	0.004202	1.000000	0.005308	0.009497	-0.014883	-0.012419	0.016972	-0.014025	0.001665	0.015271	-0.006248
CreditScore	0.005840	0.005308	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
Age	0.000783	0.009497	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
Tenure	-0.006495	-0.014883	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
Balance	-0.009067	-0.012419	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
NumOfProducts	0.007246	0.016972	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
HasCrCard	0.000599	-0.014025	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
IsActiveMember	0.012044	0.001665	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
EstimatedSalary	-0.005988	0.015271	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
Exited	-0.016571	-0.006248	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

Task-4:

Perform descriptive statistics on the dataset

Solution:

df.describe()

#----------------#

#----------------#

Screenshot:

In [119…
```
#descriptive analysis
df.describe()
```

Out[119…

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

Task-5:

Handle the Missing values.

Solution:

```
#check for null values
df.isnull().any()

df['CreditScore'].fillna(df['CreditScore'].mean(),inplace=True)
df['Age'].fillna(df['Age'].median(),inplace=True)
df['Tenure'].fillna(df['Tenure'].median(),inplace=True)
df['Balance'].fillna(df['Balance'].median(),inplace=True)
df['CreditScore'].fillna(df['CreditScore'].median(),inplace=True)
df['NumOfProducts'].fillna(df['NumOfProducts'].median(),inplace=True)
df['HasCrCard'].fillna(0,inplace=True)
df['IsActiveMember'].fillna(0, inplace=True)
df['EstimatedSalary'].fillna(df['EstimatedSalary'].mean(), inplace=True)
#-----------------#
#-----------------#
```

Screenshot:

CHECKING FOR NULL VALUES IN ANY OF THE COLUMNS

In [117…
```
df.isnull().any()
```

Out[117…
```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

This shows that there are no null values or missing values in any of the columns of the dataset.

1. HANDLING THE MISSING VALUES

For numerical columns we can use mean or median for replacing null values.

In [118…

```
df['CreditScore'].fillna(df['CreditScore'].mean(),inplace=True)
df['Age'].fillna(df['Age'].median(),inplace=True)
df['Tenure'].fillna(df['Tenure'].median(),inplace=True)
df['Balance'].fillna(df['Balance'].median(),inplace=True)
df['CreditScore'].fillna(df['CreditScore'].median(),inplace=True)
df['NumOfProducts'].fillna(df['NumOfProducts'].median(),inplace=True)
df['HasCrCard'].fillna(0,inplace=True)
df['IsActiveMember'].fillna(0, inplace=True)
df['EstimatedSalary'].fillna(df['EstimatedSalary'].mean(), inplace=True)
```

Task-6:

Find the outliers and replace the outliers

Solution:

```
#detecting outliers
sns.boxplot(df['Age'])

#replacing outliers
Q1= df['Age'].quantile(0.25)
Q3=df['Age'].quantile(0.75)
IQR=Q3-Q1
upper_limit =Q3 + 1.5*IQR
lower_limit =Q1 - 1.5*IQR
# df=df[df['Age']<upper_limit]
df['Age'] = np.where(df['Age']>upper_limit,37,df['Age']) #median 37
#-----------------#
#-----------------#
```

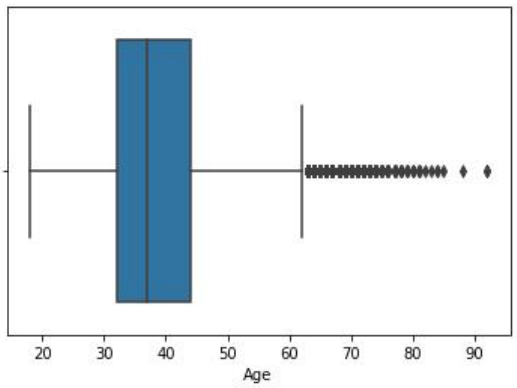
Screenshot:

In [128…

```
sns.boxplot(df['Age'])
```

Out[128…

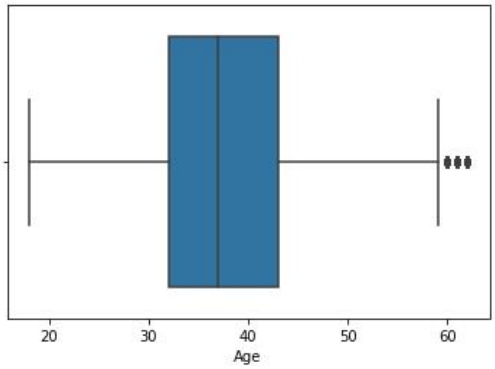
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4f77328110>
```



There are outliers in the 'Age' column values

```
In [131…  sns.boxplot(df['Age'])

Out[131…  <matplotlib.axes._subplots.AxesSubplot at 0x7f4f7735ec10>
```



Task-7:

Check for Categorical columns and perform encoding.

Solution:

```
#check for categorical columns
textualColumns = [x for x in df.columns if df[x].dtype == np.dtype('O')]
print(textualColumns)

#perform label encoding to gender column
from sklearn.preprocessing import LabelEncoder
lbEnc=LabelEncoder()
df['Gender'] = lbEnc.fit_transform(df['Gender'])

#perform one hot encoding to Geography column
df_main=pd.get_dummies(df,columns=['Geography'])
df_main_main=df_main.drop(columns=['Surname'], axis=1)
df_main_main.head(10)
#-----------------#
#-----------------#
```

Screenshots:

1. CHECK FOR CATEGORICAL COLUMNS

In [132…
```
textualColumns = [x for x in df.columns if df[x].dtype == np.dtype('O')]
print(textualColumns)
```

```
['Surname', 'Geography', 'Gender']
```

Now we drop the 'Surname' column because it is neither a numerical column nor a categorical column and is of no use in the future predictions.

In [133…
```
df.drop(columns=['Surname'],axis=1)
```

Out[133…

	RowNumber	CustomerId	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
9995	9996	15606229	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 13 columns

LABEL ENCODING is done to the categorical column 'Gender'

In [134…
```
from sklearn.preprocessing import LabelEncoder
lbEnc=LabelEncoder()
df['Gender'] = lbEnc.fit_transform(df['Gender'])
```

In [135…
```
df.head(10)
```

Out[135…

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	0	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	0	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	0	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	0	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	0	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645	Spain	1	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	1	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	376	Germany	0	29	4	115046.74	4	1	0	119346.88	1
8	9	15792365	He	501	France	1	44	4	142051.07	2	0	1	74940.50	0
9	10	15592389	H?	684	France	1	27	2	134603.88	1	1	1	71725.73	0

ONE HOT ENCODING

If an additional information is provided about the given dataset's 'Geography' column, saying all the people belonging to only 3 countries Spain, France and Germany, ONE HOT ENCODING can be performed on that column.

In [136…
```
df_main=pd.get_dummies(df,columns=['Geography'])
df_main_main=df_main.drop(columns=['Surname'], axis=1)
df_main_main.head(10)
```

Out[136…

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_France	Geography_G
0	1	15634602	619	0	42	2	0.00	1	1	1	101348.88	1	1	
1	2	15647311	608	0	41	1	83807.86	1	0	1	112542.58	0	0	
2	3	15619304	502	0	42	8	159660.80	3	1	0	113931.57	1	1	
3	4	15701354	699	0	39	1	0.00	2	0	0	93826.63	0	1	
4	5	15737888	850	0	43	2	125510.82	1	1	1	79084.10	0	0	
5	6	15574012	645	1	44	8	113755.78	2	1	0	149756.71	1	0	
6	7	15592531	822	1	50	7	0.00	2	1	1	10062.80	0	1	
7	8	15656148	376	0	29	4	115046.74	4	1	0	119346.88	1	0	
8	9	15792365	501	1	44	4	142051.07	2	0	1	74940.50	0	1	
9	10	15592389	684	1	27	2	134603.88	1	1	1	71725.73	0	1	

Task-8:

Split the data into dependent and independent variables.

Solution:

```
X=df_main_main.drop(columns=['EstimatedSalary'],axis=1)
X.head()
Y=df_main_main['EstimatedSalary']
print(Y)
#-----------------#
#-----------------#
```

Screenshots:

1. SPLITTING DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

X - independent variables

In [138…

```
X=df_main_main.drop(columns=['EstimatedSalary'],axis=1)
X.head()
```

Out[138…

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germany	Geograp
0	1	15634602	619	0	42	2	0.00	1	1	1	1	1	0	
1	2	15647311	608	0	41	1	83807.86	1	0	1	0	0	0	
2	3	15619304	502	0	42	8	159660.80	3	1	0	1	1	0	
3	4	15701354	699	0	39	1	0.00	2	0	0	0	1	0	
4	5	15737888	850	0	43	2	125510.82	1	1	1	0	0	0	

Y - dependent variable (EstimatedSalary)

In [139…

```
Y=df_main_main['EstimatedSalary']
print(Y)
```

```
0       101348.88
1       112542.58
2       113931.57
3        93826.63
4        79084.10
          ...
9995     96270.64
9996    101699.77
9997     42085.58
9998     92888.52
9999     38190.78
Name: EstimatedSalary, Length: 10000, dtype: float64
```

Task-9:

Scale the independent variables

Solution:

```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame(scale(X),columns=X.columns)
X_scaled.head()
#-----------------#
#-----------------#
```

Screenshot:

In [140…

```
from sklearn.preprocessing import scale
X_scaled=pd.DataFrame(scale(X),columns=X.columns)
X_scaled.head()
```

Out[140…

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Germar
0	-1.731878	-0.783213	-0.326221	-1.095988	0.490105	-1.041760	-1.225848	-0.911583	0.646092	0.970243	1.977165	0.997204	-0.5787
1	-1.731531	-0.606534	-0.440036	-1.095988	0.374424	-1.387538	0.117350	-0.911583	-1.547768	0.970243	-0.505775	-1.002804	-0.5787
2	-1.731185	-0.995885	-1.536794	-1.095988	0.490105	1.032908	1.333053	2.527057	0.646092	-1.030670	1.977165	0.997204	-0.5787
3	-1.730838	0.144767	0.501521	-1.095988	0.143063	-1.387538	-1.225848	0.807737	-1.547768	-1.030670	-0.505775	0.997204	-0.5787
4	-1.730492	0.652659	2.063884	-1.095988	0.605786	-1.041760	0.785728	-0.911583	0.646092	0.970243	-0.505775	-1.002804	-0.5787

Task-10:

Split the data into training and testing

Solution:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test =train_test_split(X_scaled,y, test_size=0.3,random_state=0)
print(X_train.shape)
X_train
#-----------------#
#-----------------#
```

Screenshot:

1. SPLIT THE DATA INTO TRAINING AND TESTING

In [141…

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test =train_test_split(X_scaled,y, test_size=0.3,random_state=0)
```

In [142…

```
print(X_train.shape)
X_train
```

```
(7000, 14)
```

Out[142…

	RowNumber	CustomerId	CreditScore	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	Exited	Geography_France	Geography_Ge
7681	0.928899	-0.797032	-0.098592	0.912419	-0.551023	-1.041760	1.117213	0.807737	0.646092	0.970243	1.977165	0.997204	-0.5
9031	1.396553	0.714314	-1.133270	0.912419	0.143063	0.687130	-1.225848	0.807737	0.646092	-1.030670	-0.505775	0.997204	-0.5
3691	-0.453278	0.963450	-0.626278	-1.095988	-0.088299	-0.004426	1.354191	-0.911583	-1.547768	0.970243	-0.505775	0.997204	-0.5
202	-1.661903	-1.250707	-1.391939	0.912419	1.415552	-0.004426	-1.225848	-0.911583	-1.547768	0.970243	1.977165	-1.002804	-0.5
5625	0.216680	-0.385174	-1.474714	-1.095988	2.572361	0.687130	1.070229	-0.911583	0.646092	0.970243	-0.505775	0.997204	-0.5
…	…	…	…	…	…	…	…	…	…	…	…	…	…
9225	1.463756	-1.473777	-0.584891	-1.095988	-0.666704	-0.350204	0.698607	0.807737	0.646092	0.970243	-0.505775	-1.002804	1.7
4859	-0.048671	-0.609314	1.484464	-1.095988	-1.823512	-0.350204	0.608299	-0.911583	0.646092	0.970243	-0.505775	-1.002804	-0.5
3264	-0.601195	-1.620525	0.905045	0.912419	-0.319661	-0.004426	1.358909	0.807737	0.646092	-1.030670	-0.505775	0.997204	-0.5
9845	1.678530	-0.374039	-0.626278	-1.095988	0.027382	1.378686	-1.225848	0.807737	0.646092	0.970243	-0.505775	-1.002804	-0.5
2732	-0.785485	-1.364118	-0.284834	-1.095988	1.184190	-1.387538	0.506303	-0.911583	0.646092	-1.030670	1.977165	-1.002804	1.7

7000 rows × 14 columns